JingJing Zhuang

CSE 489/589: Modern Networking Concepts (Spring 2020)

Programming Assignment 2: Reliable Transport Protocols

Instructor: Dimitrios Koutsonikolas

“I have read and understood the course academic integrity policy.”

Timeout Scheme for three reliable data transport protocols:

1. **Alternating-Bit Protocol:** I choose the timer by comparing the experimental results of different timer value in the given loss probability. Set the timer to 10.0 time unit can produce more throughput than other time value as the loss probability increase.
2. **Go-Back-N Protocol:** By comparing which timer value works for the best throughput, I use the run_experiments to test each timer value in different window size.
3. **Selective-Repeat Protocol:** Since the SR protocol need to set up timers for each individually packets, I will be more focusing on the loss probability. During analyzing the result of SR protocol with different timer value, I use the given loss probability to determine which timer works for the best.

Selective-Repeat Protocol:

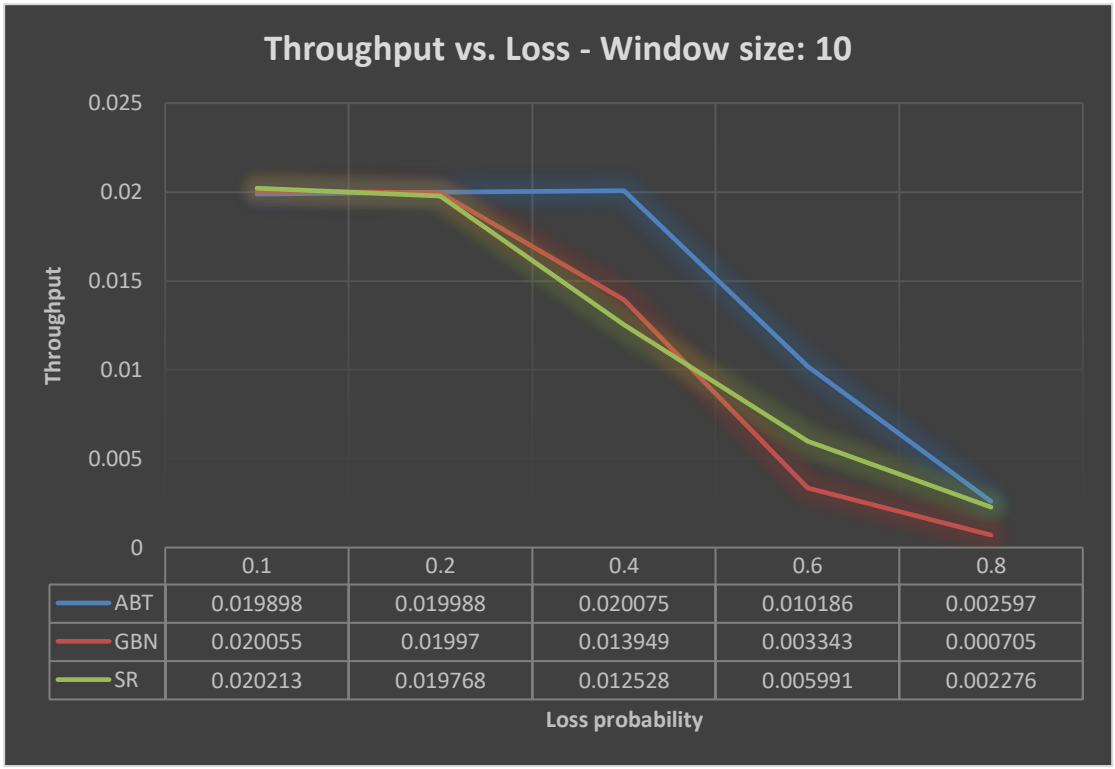
The idea that I implemented multiple timers in SR using one hardware time is that I set up a timer (use get_sim_time()) for every packet that is transmitting and use a vector called record_seq to store the seq number of each packet that has a timer on. When a packet arrives for layer 4 (A_input), first it would check the checksum of the packet, if the checksum is valid, then it would check is the packet seq number in the window, if it is true then mark the packet as acked. Next, check is the packet’s acknum equal to the send base, if they are equal, then move the send base to next unacked seq number and send the next seq number packet that is in the window and start timer for that packet, point next seq to next location. Now, for stopping a timer, I would check the seq number in the record_seq (use to store the seq number of each packet that has a timer on.). If the earliest seq number in the record_seq equals to the packet’s acknum, then stop the timer, and remove all the acked seq # in the record list. If the record_seq list is not empty, then calculate the next timeout for the earliest unacked packet in the record_seq list, and starts a timer with the next timeout timer.

Performance comparison:

In each of the following 2 experiments, run each of the protocols with a total number of 1000 messages to be sent by entity A, a mean time of 50 between message arrivals (from A's layer5) and a corruption probability of 0.2.

- **Experiment 1:**

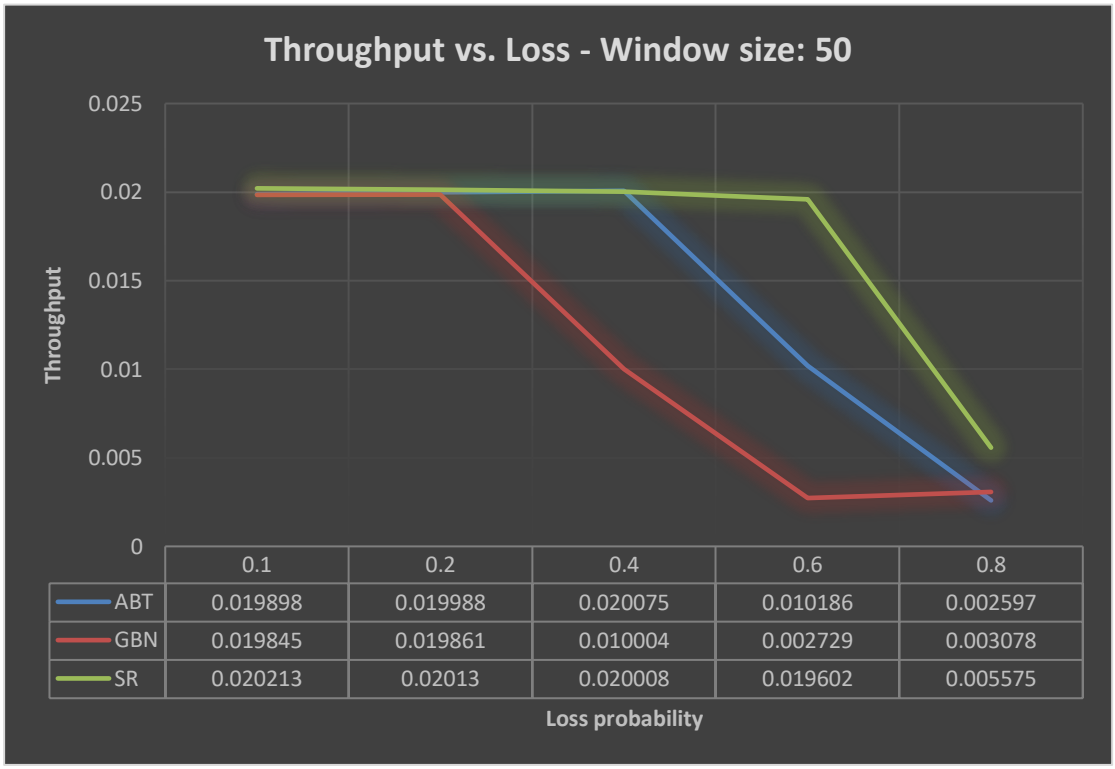
  With loss probabilities: {0.1, 0.2, 0.4, 0.6, 0.8}, compare the 3 protocols' throughputs at the application layer of receiver B. Use 2 window sizes: {10, 50} for the Go-Back-N version and the Selective-Repeat Version.



	0.1	0.2	0.4	0.6	0.8
ABT	0.019898	0.019988	0.020075	0.010186	0.002597
GBN	0.020055	0.01997	0.013949	0.003343	0.000705
SR	0.020213	0.019768	0.012528	0.005991	0.002276

Observation:

- As the loss probability increase, ABT protocol has higher throughput than other two protocols as the loss probability increase.
- Three protocols has similar throughput when loss possibility equal to 0.1 and 0.2
- As the loss probability increase, three protocols' throughput are decreased, and SR protocol's throughput slowly overtook GBN protocols.



	0.1	0.2	0.4	0.6	0.8
ABT	0.019898	0.019988	0.020075	0.010186	0.002597
GBN	0.019845	0.019861	0.010004	0.002729	0.003078
SR	0.020213	0.02013	0.020008	0.019602	0.005575

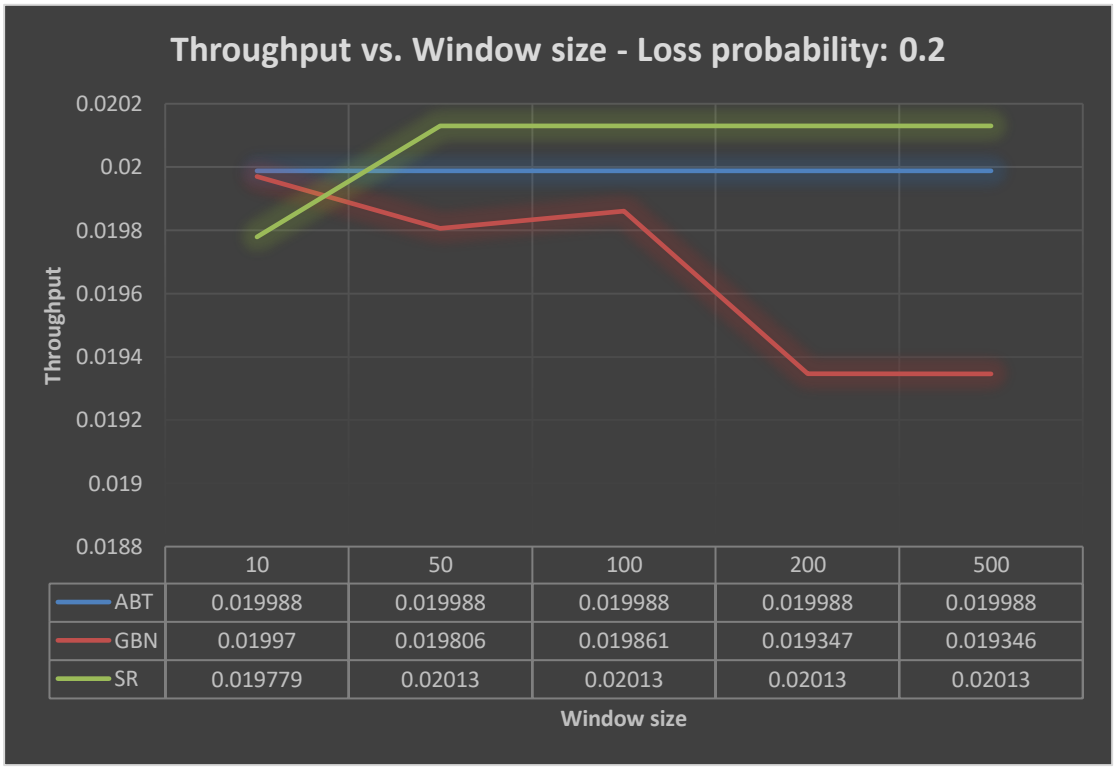
Observation:

- According to this graph, SR has higher throughput than other two protocols as the loss probability increases.
- GBN's throughput decreases as the window size grows
- ABT data remain the same since it did not need the window size

I expect the throughput would decrease as the loss probability increase, because as the loss probability increase, the retransmission rate are also increase. Especially for GBN and SR, GBN need to be retransmit all the packet from base to next seq number, and SR need to resend all the unacked packet. So I agree with the measurements. In higher loss probability, SR is more efficient than the GBN because SR only need to resend the packet that are not acked, but GBN are not. Also, since the window size increase from 10 to 50, the throughput for GBN decreases, this is due to the range of resend packet increase, since when there's a timeout occur, GBN need to resend the packet from base to next seq number. I did not agree with the result for SR in window size 10 experiment, because the change in window size shouldn't affect much for SR protocol, because it only resend the packet that are loss/unacked.

- **Experiment 2:**

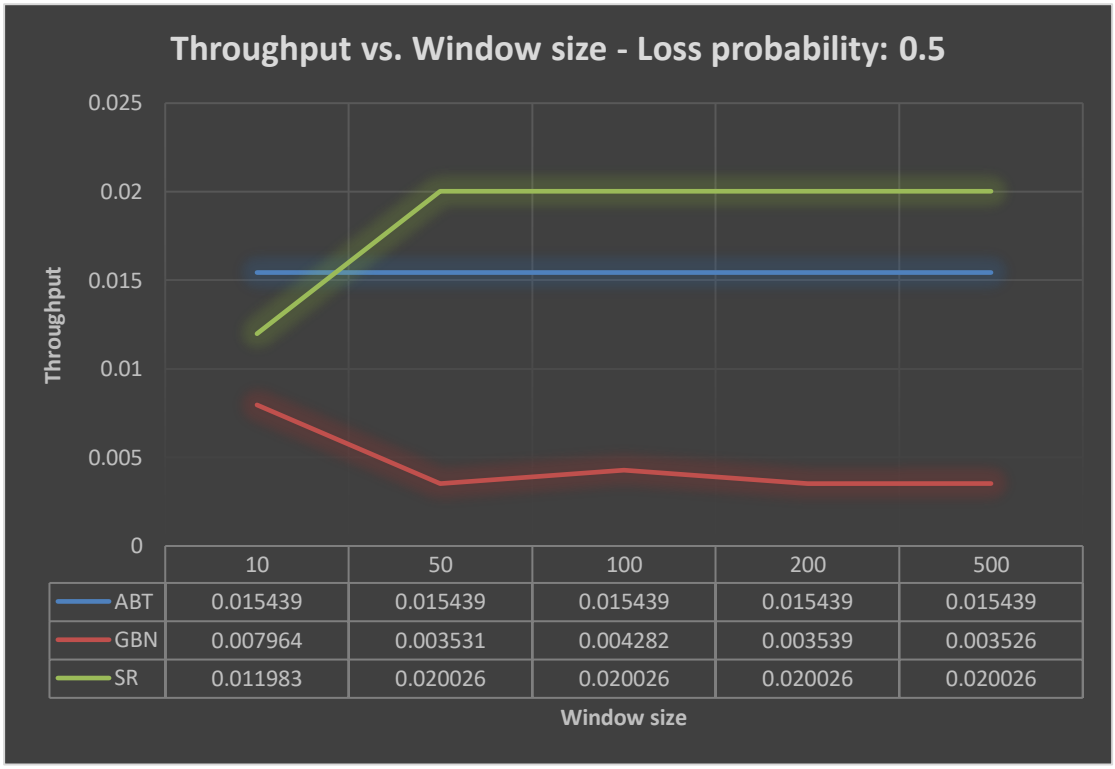
  With window sizes: {10, 50, 100, 200, 500} for GBN and SR, compare the 3 protocols' throughputs at the application layer of receiver B. Use 3 loss probabilities: {0.2, 0.5, 0.8} for all 3 protocols.



	10	50	100	200	500
ABT	0.019988	0.019988	0.019988	0.019988	0.019988
GBN	0.01997	0.019806	0.019861	0.019347	0.019346
SR	0.019779	0.02013	0.02013	0.02013	0.02013

Observation:

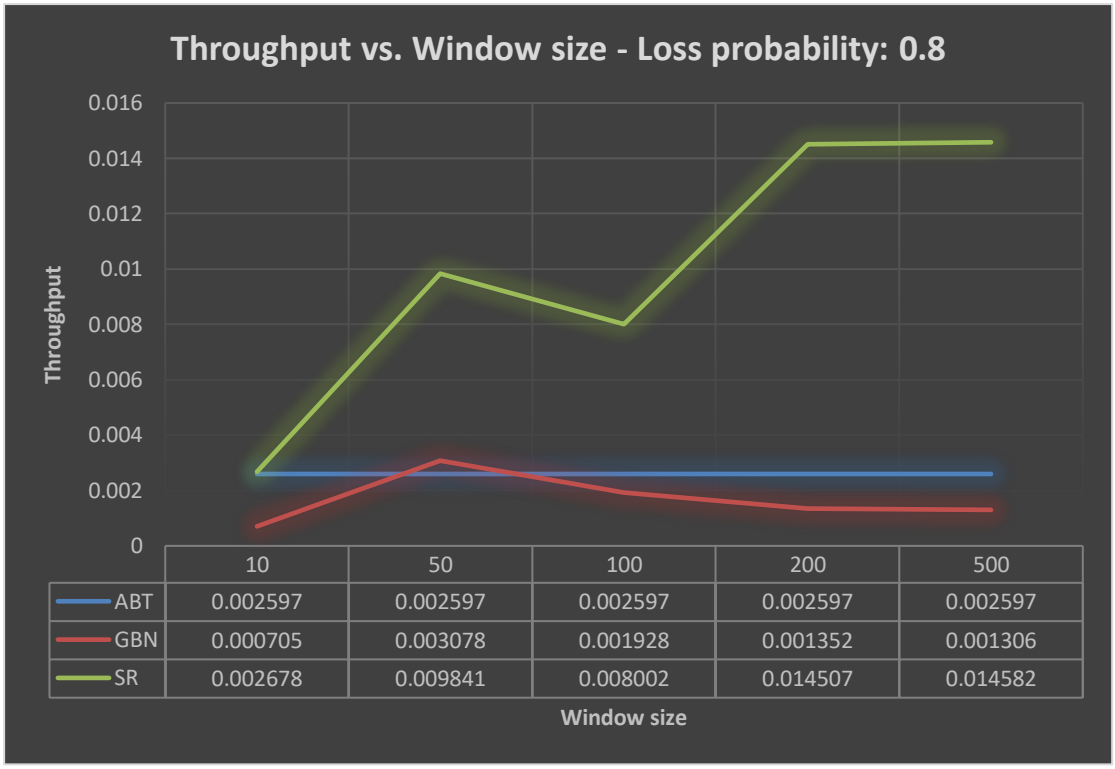
- According to this graph, ABT has constant throughput as the window size grows larger, since it do not need the window size.
- GBN protocol's throughput decrease as the window size increase, GBN protocol performs pretty good when the window size is small.
- In constant 0.2 loss probability, SR protocol's throughput does not change much as the window size grows. It has low throughput at window 10, then it grows constantly afterward.



	10	50	100	200	500
ABT	0.015439	0.015439	0.015439	0.015439	0.015439
GBN	0.007964	0.003531	0.004282	0.003539	0.003526
SR	0.011983	0.020026	0.020026	0.020026	0.020026

Observation:

- According to this graph, ABT has constant throughput as the window size grows larger, since it do not need the window size. But ABT protocol's throughput decrease as the loss probability increase from 0.2 to 0.5
- As the window size increase, GBN throughput decrease. The time for GBN protocol to complete transmission takes longer.
- SR has low throughput at window 10, then it grows constantly afterward.



	10	50	100	200	500
ABT	0.002597	0.002597	0.002597	0.002597	0.002597
GBN	0.000705	0.003078	0.001928	0.001352	0.001306
SR	0.002678	0.009841	0.008002	0.014507	0.014582

Observation:

- According to this graph, ABT has constant throughput as the window size grows larger, since it do not need the window size. But ABT protocol's throughput decrease as the loss probability increase from 0.5 to 0.8
- GBN protocol perform inefficiently in large window size and high loss probability condition.
- SR protocol's throughput has an upward trend as window size grows and loss probability are high.

I expect the throughput for three protocols be decrease when the loss probability increases {0.2, 0.5, 0.8}, and the throughput for ABT and SR remain constant when loss probability is constant and window size increases. Because the window size doesn't need for ABT, and for SR, the throughput did not change much when the window size equal {0.2, 0.5} because when SR experiencing a timeout, it would only resend the unacked packet, and receiver individually acknowledges all correctly received pkts. But when window size equal 0.8, the SR starts increasing when the window size grows, this might due to the higher resend rate and since the window size grows, SR can send more unacked packet so they can get acked. Unlike the GBN, complete transmission would take longer when window size increase, because the range of packet that need to resend increase. Also, the increase of loss probability would affect the throughput of three protocols, this is because when the loss probability increase the retransmission rate also increases.